Routing Foundation Model (RFM)

A Unified Neural Optimization Framework
for Large-Scale Routing and MILPs

Ritwika Kancharla
ritwikareddykancharla@gmail.com

December 6, 2025

Abstract

Large-scale routing and supply-chain systems such as Amazon's middle-mile and last-mile networks are routinely modeled as mixed-integer linear programs (MILPs) with tens of thousands of binary variables and tight operational constraints. Classical solvers provide high-quality solutions but are often too slow for real-time re-optimization, while existing Neural Combinatorial Optimization models do not explicitly encode MILP structure or constraint geometry. This monograph proposes the *Routing Foundation Model (RFM)*, a unified neural optimization framework that treats transformer-style architectures as learned surrogate solvers for routing MILPs. RFM combines (i) MILP-aware encoders, (ii) constraint-specialized Mixture-of-Experts, (iii) dual-informed attention via violation signals $Ax - b$, (iv) diffusion-style generative priors over discrete routing decisions, and (v) world-model components for multi-step logistics planning. We formulate the framework, connect it to classical primal–dual and proximal optimization, and outline experimental protocols and open problems toward foundation models for routing and supply-chain optimization.

Contents

1 Introduction

Modern logistics networks—including those operated by Amazon, UPS, DHL, and national postal systems—form some of the most complex engineered systems in the world. These networks must repeatedly solve large-scale, tightly constrained routing problems involving transportation links, facility capacities, shipment flows, and strict service-level agreements (SLAs). Solving such problems reliably, efficiently, and under stringent latency requirements is essential for operational excellence yet remains one of the hardest challenges in combinatorial optimization Wolsey [1999], Bertsimas and Tsitsiklis [1997].

Traditional mixed-integer linear programming (MILP) formulations provide a principled mathematical foundation for routing problems. They encode discrete decisions, coupling constraints, and objective functions in a unified structure amenable to branch-and-bound, cutting planes, and primal–dual heuristics. Modern solvers such as Gurobi, CPLEX, and SCIP are extraordinarily powerful, but their runtime frequently increases sharply with instance size, making repeated optimization infeasible for real-time decision-making in dynamic supply-chain environments Nair et al. [2020], Lodi and Zarpellon [2017]. Even when warm-started, large MILPs often exceed the available latency budget.

In parallel, neural combinatorial optimization (Neural CO) has emerged as a compelling paradigm. Sequence-to-sequence architectures such as pointer networks Vinyals et al. [2015], transformer-based routing solvers Kool et al. [2019], and reinforcement-learning approaches for VRP Nazari et al. [2018], Bello et al. [2016] have demonstrated the ability to produce feasible routing solutions with extremely low runtime at inference. These models show that learned heuristics can outperform classical handcrafted heuristics under certain conditions. However, existing Neural CO approaches typically do not encode the algebraic structure of MILPs, lack explicit dual reasoning, and often generalize poorly outside their training distributions Joshi et al. [2020], Bengio et al. [2021].

This monograph introduces the **Routing Foundation Model (RFM)**, a unified neural optimization framework that integrates the strengths of classical MILP solving with modern deep learning architectures. RFM treats transformer-style networks not simply as policy generators but as *learned surrogate solvers* whose internal layers approximate primal–dual reasoning, constraint interactions, and refinement steps. By embedding MILP structure directly into attention mechanisms, refinement layers, and Mixture-of-Experts (MoE), RFM provides a scalable, structure-aware backbone for large-scale routing optimization.

1.1 Routing at Industrial Scale

Large-scale routing networks exhibit four defining characteristics:

- **High dimensionality:** Middle-mile and last-mile routing problems can involve tens or hundreds of thousands of binary variables describing flows, activation decisions, and feasible transitions Chen et al. [2021].
- **Dynamic environments:** Demand surges, congestion, weather variation, and facility disruptions require models that can adapt quickly to new conditions.
- **Hard feasibility constraints:** Violations of capacity, precedence, or timing constraints are operationally unacceptable, making feasibility a first-class requirement.
- **Strict latency constraints:** Inference must often complete within milliseconds to seconds.

These constraints collectively exceed the capability of classical optimization techniques when scaled to real-time, high-frequency decision-making.

1.2 MILPs, Solvers, and Latency Bottlenecks

Routing problems are frequently modeled as MILPs:

$$\min_x \; c^\top x \quad \text{s.t.} \quad Ax \leq b, \;\; x \in \{0,1\}^n,$$

where x encodes routing decisions and A encodes feasibility and capacity constraints. MILPs offer provable guarantees and powerful modeling abstractions Wolsey [1999], but exhibit the following challenges in large-scale settings:

- computational cost can grow exponentially in the worst case;
- presolve and branching heuristics may still exceed latency budgets;
- solver warm-starting remains expensive under large perturbations;
- generating many feasible alternatives is slow.

Recent work on integrating learning inside solvers—e.g., learning to branch Gasse et al. [2019], Lodi and Zarpellon [2017] or learning warm-starts Nair et al. [2020]—has shown promise but still depends on a heavy solver backend.

1.3 From Neural CO to Neural Surrogate Solvers

Transformer-based neural solvers have demonstrated the potential of sequence modeling in routing tasks Kool et al. [2019]. Reinforcement-learning methods Nazari et al. [2018], Bello et al. [2016] and GNN-based policies Joshi et al. [2020] improve generalization but still lack explicit representations of constraint families, dual variables, or KKT structure. As a consequence, they often:

- fail to generalize to new operational constraints;

- provide no certificate of feasibility;
- degrade unpredictably under perturbations;
- lack robustness guarantees.

RFM proposes a different paradigm: treat neural models as *approximate optimization layers* inspired by differentiable optimization Amos and Kolter [2017], Pogančić et al. [2020] and implicit-layer modeling Blondel et al. [2022]. This enables transformers to approximate dual updates, constraint interactions, and proximal refinement—bridging the gap between classical optimization and deep learning.

1.4 Goals and Scope of This Monograph

This monograph advances the foundations of Routing Foundation Models by:

- formalizing routing MILPs and their neural approximations;
- introducing the **MILP-Transformer**, a differentiable surrogate solver inspired by primal–dual optimization;
- defining the **Neural Routing Optimization Model (NROM)**, an architecture informed by MILP structure and graph topology;
- proposing diffusion priors Nichol and Dhariwal [2021], Hoogeboom et al. [2021] for modeling distributions over feasible routing patterns;
- framing routing as a world-model learning problem Ha and Schmidhuber [2018], Hafner et al. [2019].

These components combine to form a scalable and generalizable foundation model for routing and supply-chain optimization.

Contributions

The contributions of this monograph are:

- A unified framework integrating MILP structure with transformer-based neural architectures.
- The MILP-Transformer: a surrogate solver incorporating constraint embeddings, dual-informed attention, and refinement layers.
- The NROM architecture: a structure-aware routing policy grounded in MILP variables, constraints, and graph representations.
- Diffusion priors for generating feasible routing configurations.
- World-model architectures enabling multi-step routing and planning.

2 Background: Routing, MILPs, and Neural Optimization

Routing and network optimization problems form a central class of combinatorial optimization problems arising in logistics, transportation, and supply-chain systems. This chapter reviews the key mathematical foundations relevant to the Routing Foundation Model (RFM), including routing formulations, mixed-integer linear programming (MILP), neural combinatorial optimization, and recent approaches that integrate machine learning with classical solvers. Together, these concepts define the landscape in which RFM operates.

2.1 Routing and Network Optimization Problems

Routing problems involve determining the movement of vehicles, shipments, or flows across a network while satisfying operational constraints such as capacity, timing, distance, and precedence. Many such problems fall into the broad class of NP-hard combinatorial problems and have been extensively studied in operations research.

2.1.1 Vehicle Routing Problems

The Vehicle Routing Problem (VRP) and its variants—including capacitated VRP, VRP with time windows (VRPTW), pickup-and-delivery, and multi-depot VRP—serve as foundational abstractions for transportation systems. These problems often require constructing one or more tours that satisfy capacity and demand constraints while minimizing total travel cost.

Neural solution approaches to routing tasks gained attention with pointer networks Vinyals et al. [2015] and later transformer-based solvers Kool et al. [2019]. While these models excel on Euclidean routing benchmarks, they typically lack structural features needed in real logistics systems, such as flow conservation or facility-level constraints.

2.1.2 Middle-Mile and Last-Mile Logistics

Industrial routing systems for e-commerce and parcel delivery extend far beyond classical VRP. Middle-mile logistics connects fulfillment centers, sorting centers, and distribution nodes via

scheduled or dynamically generated transportation links. Last-mile routing involves thousands of delivery units, complex timing constraints, and volatile demand patterns.

Unlike classical routing benchmarks, real networks exhibit:

- heterogeneous node types (e.g., FCs, SCs, DSs),
- multi-commodity flows,
- strict precedence and SLA constraints,
- highly dynamic state (traffic, delays),
- massive scale (orders of magnitude larger than VRP instances).

These factors make MILP formulations both expressive and indispensable, but also computationally challenging.

2.2 Mixed-Integer Linear Programming (MILP)

MILPs provide a general modeling framework for routing and logistics problems. An MILP consists of a linear objective function and linear constraints, with some or all decision variables restricted to be integer-valued:

$$\min_x \; c^\top x \quad \text{s.t.} \quad Ax \le b, \quad x \in \{0,1\}^n.$$

Here, the matrix A captures structural requirements such as flow conservation, capacity limits, time-window feasibility, and precedence relationships. The cost vector c encodes objectives such as distance, delay penalties, or congestion costs.

MILPs remain the gold standard for high-fidelity routing optimization Wolsey [1999], Bertsimas and Tsitsiklis [1997]. However, classical solvers face inherent challenges:

- combinatorial explosion of the search space,
- large branch-and-bound trees,
- expensive cut generation,
- unpredictable latency on real-world instances.

Even with decades of optimization research, large routing MILPs remain difficult to solve within tight operational time budgets.

2.2.1 Primal–Dual and Lagrangian Views

Many solver techniques rely on primal–dual reasoning, where violation signals $v = \max(0, Ax - b)$ and their dual multipliers inform branching, cutting, and heuristic decisions. Lagrangian relaxation provides an alternative view:

$$\mathcal{L}(x, \lambda) = c^\top x + \lambda^\top (Ax - b),$$

which leads to iterative updates resembling gradient ascent in the dual domain and heuristic

search in the primal domain. These conceptual tools inspire several components of the MILP-Transformer introduced later.

2.3 Neural Combinatorial Optimization

Neural Combinatorial Optimization (Neural CO) seeks to approximate solutions to NP-hard problems through learned heuristics. Pointer networks Vinyals et al. [2015] were the first sequence models designed to construct combinatorial structures. Subsequent works used reinforcement learning Bello et al. [2016], Nazari et al. [2018] and transformers Kool et al. [2019] to improve generalization and construction quality.

Despite strong empirical performance on benchmark tasks, Neural CO models face several limitations:

- **Weak constraint handling:** They do not model MILP structure explicitly, making feasibility brittle.
- **Limited transferability:** They struggle to generalize across network sizes, topologies, and constraint sets Joshi et al. [2020].
- **Lack of interpretability:** Their internal states do not correspond to optimization primitives such as dual variables.

These issues arise because Neural CO models primarily operate in the policy or sequence space rather than in the constraint space.

2.4 Deep Learning for MILPs (DL4MIP)

Recent work attempts to integrate learning directly into MILP pipelines. Examples include:

- **Learning branching decisions** GNN-based models predict branching priorities in branch-and-bound solvers, improving search efficiency Gasse et al. [2019], Lodi and Zarpellon [2017].
- **Learning warm-starts** Neural models predict feasible or near-feasible initial solutions to reduce solver runtime Nair et al. [2020].
- **Differentiable optimization layers** OptNet Amos and Kolter [2017] and differentiable LP layers Pogančić et al. [2020] enable embedding optimization steps into neural architectures.
- **Implicit layers and differentiable solvers** Tools such as efficient implicit differentiation Blondel et al. [2022] allow optimization routines to be treated as differentiable implicit functions.

While powerful, these approaches usually require a classical solver for full resolution and do not generalize to large, highly structured routing instances without solver intervention.

2.5 Limitations of Current Approaches

The methods above reveal several fundamental barriers:

- Classical MILPs are accurate but slow.
- Neural CO is fast but fails under structural or distributional shift.
- Existing hybrid MILP-ML solutions depend heavily on solver infrastructure.

This gap motivates the development of RFM: a fully neural, structure-aware surrogate solver that approximates MILP reasoning while remaining fast and generalizable.

2.6 Positioning RFM Within This Landscape

The Routing Foundation Model aims to unify three traditions:

1. **MILP structure and primal–dual optimization**, providing feasibility and interpretability.
2. **Transformer architectures**, providing expressive, scalable reasoning capabilities.
3. **Generative and world-model approaches**, capturing the distribution of routing configurations and multi-step dynamics.

By embedding MILP structure directly into transformer computations, RFM seeks to inherit the strengths of both classical optimization and modern deep learning, enabling fast, feasible, and robust routing decisions at industrial scale.

3 Transformers Through the Lens of Optimization

Transformers have emerged as the dominant architecture for sequence modeling, graph reasoning, and large-scale foundation models. While the standard view interprets transformers as powerful attention-based function approximators, recent theoretical work suggests that they can also be viewed through the lens of optimization dynamics. This chapter explores these connections and motivates their use as surrogate solvers for routing MILPs within the Routing Foundation Model (RFM).

We examine five key perspectives:

1. transformers as iterative computation graphs,
2. attention as a learned proximal or averaging operator,
3. residual blocks as approximate gradient or descent steps,
4. the transformer stack as an implicit equilibrium system,
5. dual interpretations of constraint signals in MILP-aware models.

Together, these viewpoints reveal that transformers are not merely sequence models: they are expressive *optimization engines* whose layers can approximate the iterative structure of classical primal–dual algorithms.

3.1 Transformers as Iterative Computation

Each transformer layer performs a structured computation:

$$h^{(k+1)} = h^{(k)} + \mathrm{FFN}\Big(\mathrm{Attn}(h^{(k)})\Big),$$

which is a residual update on hidden states $h^{(k)}$. This resembles an iterative update of the form:

$$x_{t+1} = x_t + g(x_t),$$

common in optimization algorithms such as gradient descent, proximal methods, and fixed-point iterations. Deep networks with residual connections can be interpreted as unrolled optimization algorithms, with each layer performing a refinement step on internal state estimates.

In the context of routing MILPs, hidden states can represent:

- variable embeddings,
- constraint embeddings,
- dual violation signals,
- latent representations of routing decisions.

This suggests that transformers can be trained to mimic optimization trajectories, leading naturally to surrogate solver behavior.

3.2 Attention as Proximal-Like Averaging

Self-attention computes:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

This operator has two important optimization-related interpretations:

1. A learned averaging / smoothing operator

The softmax normalizes weights into a probability simplex, producing a convex combination of values V. This resembles proximal or smoothing operations of the form:

$$x_{t+1} = \sum_i \alpha_i x_i,$$

where α belongs to a simplex constraint set. Such smoothing is key in many primal–dual and proximal splitting algorithms.

2. A learned similarity metric

The dot-product attention score approximates:

$$\langle \nabla\phi(x_i), \nabla\phi(x_j) \rangle,$$

where ϕ is a learned embedding. This mirrors optimization methods that cluster or group variables based on gradient similarity or constraint structure.

Under RFM, attention is modified to incorporate constraint violations, making it closer to dual-informed proximal operators.

3.3 Residual Blocks as Learned Descent Steps

A transformer layer can be decomposed as:

$$h^{(k+1)} = h^{(k)} + \underbrace{f\big(h^{(k)}\big)}_{\text{descent-like update}},$$

where f represents attention + feedforward transformations. In optimization algorithms, many updates take the form:

$$x_{t+1} = x_t - \eta \nabla_x \mathcal{L}(x_t),$$

or more generally:

$$x_{t+1} = x_t + \Delta_t,$$

where Δ_t may incorporate projections, constraints, or Lagrangian terms.

Transformers approximate such update rules with learned descent directions derived from:

- pairwise relationships between variables,
- constraint interactions,
- dual signals or violation magnitudes,
- global routing context.

This makes residual transformer blocks a natural fit for modeling iterative refinement in MILP surrogate solving.

3.4 Implicit Layers and Deep Equilibrium Views

Transformers can also be viewed as solving for an implicit fixed point:

$$h^* = h^* + f(h^*),$$

or equivalently:

$$f(h^*) = 0,$$

which is characteristic of deep equilibrium (DEQ) models. Such implicit layers approximate solutions to nonlinear systems or optimality conditions Blondel et al. [2022].

Under this perspective:

- the transformer stack approximates a converged optimizer,

- training encourages layers to move toward a fixed point,
- internal representations approximate primal–dual equilibria.

For routing MILPs, this implies that transformers can learn to approximate KKT-like stationary conditions over discrete variables when trained appropriately.

3.5 Dual Ascent Interpretation of Constraint Signals

In RFM, the MILP-Transformer module includes dual-informed features such as violation signals:

$$v = \max(0, Ax - b),$$

and their projected influence on variables:

$$h = A^\top v.$$

These resemble dual gradients in Lagrangian optimization:

$$\lambda_{t+1} = \lambda_t + \alpha(Ax_t - b).$$

When these signals are incorporated into attention heads, they modify the effective similarity metric and the resulting update direction. This produces layers that behave like:

- learned dual ascent steps,
- constraint-aware smoothing operators,
- projections toward feasible regions.

This interpretation provides a foundational justification for embedding MILP structure directly into transformer computations.

3.6 Implications for Neural MILP Surrogates

Viewing transformers through the optimization lens yields several insights that guide the architecture of RFM:

- **Transformers naturally support iterative refinement.** Each layer resembles a descent step.
- **Attention performs structured interactions.** These interactions can encode constraint structure when augmented with MILP signals.
- **Dual messages can be embedded directly into attention.** This connects neural reasoning to primal–dual MILP solvers.
- **Implicit equilibrium views align with KKT conditions.** Transformers can approximate solutions in an amortized manner.

These insights form the conceptual basis of the MILP-Transformer architecture described in later chapters and motivate the broader design of the Routing Foundation Model.

4 Problem Formulation: Routing MILPs at Scale

Large-scale routing and logistics networks can be abstracted as directed graphs equipped with flow, timing, and capacity constraints. Industrial middle-mile and last-mile systems introduce additional structure such as heterogeneous facility types, precedence relationships, and tightly coupled operational constraints. This chapter formalizes these components and presents the mixed-integer linear programming (MILP) formulation that underlies the Routing Foundation Model (RFM).

4.1 Network Model and Decision Variables

Let the logistics network be represented as a directed graph:

$$G = (V, E),$$

where:

- V is the set of nodes (facilities such as FCs, SCs, DSs),
- $E \subseteq V \times V$ is the set of feasible transportation arcs.

We consider two families of decision variables:

Arc-activation variables. For each arc $(i, j) \in E$, we define:

$$x_{ij} \in \{0, 1\},$$

indicating whether a vehicle or shipment traverses the arc.

Flow / load variables. For each arc (i, j):

$$f_{ij} \geq 0,$$

captures the quantity transported along that arc, e.g., package volume, weight, or number of containers.

Timing variables. For time-sensitive routing, we include:

$$t_i \in \mathbb{R}_{\geq 0},$$

representing arrival or processing time at facility i.

Additional variables. Depending on the operational model:

- vehicle assignment variables,
- service-level slack variables,
- capacity utilization indicators,
- route-length or deviation penalty terms.

These variables collectively encode the routing decisions needed for industrial-scale optimization.

4.2 Routing MILP: Full Formulation

A generic routing MILP for middle-mile or last-mile operations can be written as:

$$\begin{aligned}
\min_{x,f,t} \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{(i,j)\in E} \phi(f_{ij}) + \sum_{i\in V} \psi(t_i) \\
\text{s.t.} \quad & \sum_{j:(i,j)\in E} x_{ij} = \sum_{k:(k,i)\in E} x_{ki} && \forall i \in V && \text{(flow conservation)} \\
& f_{ij} \leq u_{ij}x_{ij} && \forall (i,j) \in E && \text{(capacity)} \\
& t_j \geq t_i + \tau_{ij} - M(1 - x_{ij}) && \forall (i,j) \in E && \text{(timing / precedence)} \\
& x_{ij} \in \{0,1\},\ f_{ij} \geq 0,\ t_i \geq 0.
\end{aligned}$$

The terms $\phi(f_{ij})$ and $\psi(t_i)$ may encode penalty functions representing congestion, SLA slack, or lateness.

The model captures the essential structure of routing optimization but may be extended with:

- multi-commodity flows,
- routing schedules or shifts,
- limits on vehicle count,
- stochastic or adversarial demand models.

Industrial MILPs often include thousands of constraints and tens of thousands of decision variables, making direct real-time optimization infeasible.

4.3 Compact Form: $Ax \leq b,\ x \in \{0,1\}^n$

To facilitate analysis and integration with neural surrogate solvers, we express the routing MILP in compact matrix form:

$$\min_x \ c^\top x \quad \text{s.t.} \quad Ax \leq b, \ \ x \in \{0,1\}^n.$$

Here:

- $x \in \mathbb{R}^n$ concatenates all binary variables (activation, routing, timing indicators),
- A encodes flow, precedence, capacity, and timing constraints,
- b represents right-hand-side capacities, times, and limits,
- c encodes routing costs, penalties, and other objectives.

This compact form is *structurally expressive*: all relevant constraints can be embedded into a single linear system.

It is also *neural-friendly*: RFM uses this representation to construct variable embeddings, constraint embeddings, dual violation signals, and attention masks in the MILP-Transformer.

4.4 MILP as a Bipartite Interaction Graph

The constraint matrix A defines a bipartite graph between:

$$\text{constraints} \quad \leftrightarrow \quad \text{variables},$$

where:

- each row of A corresponds to a constraint node,
- each column corresponds to a variable node,
- edges encode nonzero coefficients.

This view is useful because:

- many optimization algorithms propagate messages along this graph,
- GNNs can approximate primal–dual propagation,
- transformers can encode variable–constraint relationships through attention heads.

RFM directly leverages this factor-graph perspective by embedding both constraint nodes and variable nodes, enabling dual-informed attention.

4.5 Scalability and Operational Requirements

Real industrial MILPs introduce several scalability challenges:

High dimensionality. Middle-mile networks may contain:

- 10,000+ arcs,
- 100,000+ binary decision variables,
- thousands of constraints related to flow, timing, and capacity.

Operational latency. Re-optimizing every few seconds requires near-instant inference. Classical solvers often exceed allowable runtimes.

Generalization. Routing networks evolve:

- new facilities open,
- new restrictions appear,
- seasonal changes shift demand,
- disruptions modify topology.

Models must generalize to unseen instances without retraining.

Feasibility reliability. Neural CO models often produce infeasible solutions when constraints tighten. RFM counters this by:

- encoding constraints explicitly,
- incorporating violation signals $v = \max(0, Ax - b)$,
- introducing constraint-aware Mixture-of-Experts,
- refining solutions through latent proximal steps.

4.6 Summary

This chapter established the mathematical structure of routing MILPs and highlighted the scale, constraints, and complexity that make industrial routing difficult. These formulations motivate the need for a neural surrogate solver capable of:

- representing MILP structure,
- processing large, heterogeneous networks,
- approximating primal–dual reasoning,
- producing fast, feasible solutions.

The next chapter introduces the Routing Foundation Model (RFM) and its component architectures designed to address these challenges.

5 Routing Foundation Model (RFM): Architecture

The Routing Foundation Model (RFM) is designed as a unified neural architecture that integrates classical optimization structure with modern deep learning components to enable fast, feasible, generalizable routing decisions at scale. RFM combines MILP-aware encoders, a transformer-based surrogate solver, constraint-specialized Mixture-of-Experts (MoE), diffusion priors, and world-modeling components. Together, these elements form a coherent system capable of approximating primal–dual optimization, capturing distributions over routing decisions, and reasoning about multi-step network dynamics.

This chapter presents the overall architectural design, its guiding principles, and the interactions between components.

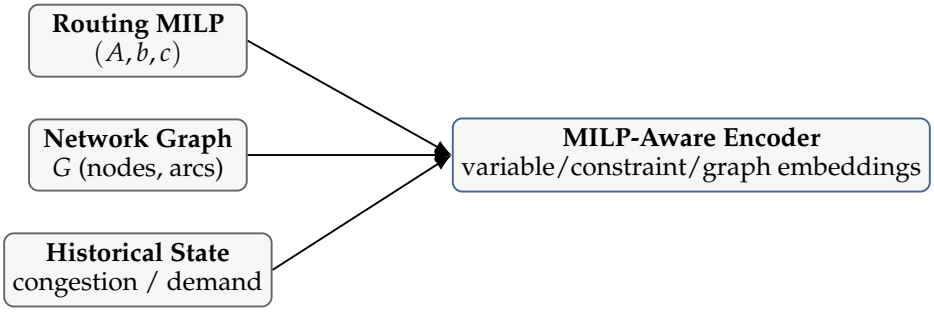

Figure 5.1: Input components of the Routing Foundation Model (RFM) and the MILP-aware encoder.

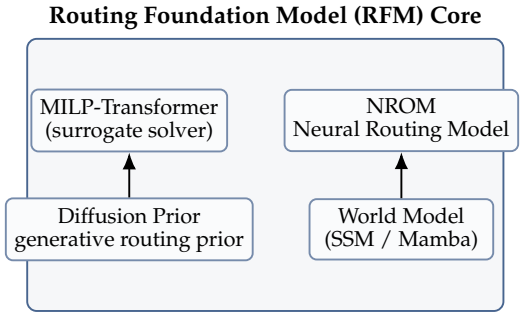

Figure 5.2: Core components of RFM: MILP-Transformer, NROM, diffusion prior, and world model.

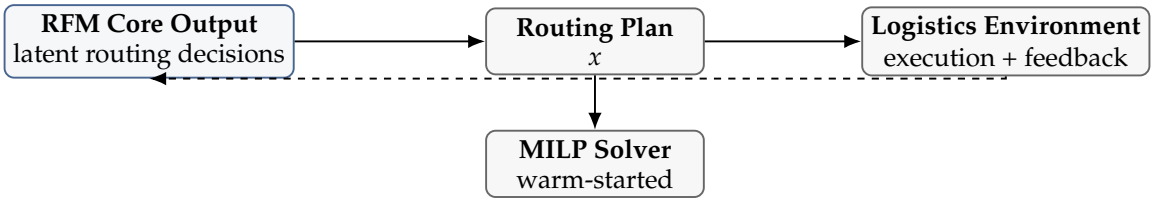

Figure 5.3: RFM output routing plans, solver warm-starting, and environment feedback loop.

5.1 Design Principles

RFM is guided by five central principles:

1. Structural awareness. The architecture encodes MILP structure directly through variable and constraint embeddings, enabling the model to operate in the same algebraic domain as classical solvers.

2. Optimization-inspired computation. Transformer blocks are interpreted as iterative optimization layers that refine routing decisions via dual-informed updates, proximal-like refinements, and constraint-aware attention mechanisms.

3. Feasibility preservation. Dual violation signals, constraint MoE modules, and latent refinement loops ensure solutions remain close to feasibility during inference.

4. Generalization across networks. RFM is designed to transfer across different network topologies, facility configurations, and constraint families by learning in the space of MILP structure rather than raw sequences.

5. Multi-modal modeling of routing decisions. Diffusion priors and world models capture stochasticity, distributional uncertainty, and multi-step dynamics in routing systems.

These principles collectively define a foundation model for routing that operates beyond single-instance heuristics.

5.2 Core Components of RFM

The RFM architecture consists of five interconnected modules:

1. MILP-aware encoder for variables and constraints,
2. MILP-Transformer surrogate solver,
3. Neural Routing Optimization Model (NROM),
4. Diffusion routing prior,

5. Routing world model.

We describe each component in detail.

5.2.1 MILP-Aware Encoder

The encoder transforms the compact MILP representation into learned embeddings usable by transformers and diffusion models. Given:

$$\min_x c^\top x \quad \text{s.t.} \quad Ax \le b, \;\; x \in \{0,1\}^n,$$

the encoder constructs:

- **Variable embeddings** Each variable x_i has an embedding v_i derived from:
  - cost coefficient c_i,
  - entries $(A_{:,i})$ showing its involvement in constraints,
  - graph features from the underlying routing network,
  - type indicators (arc variable, slack variable, timing var).
- **Constraint embeddings** For each constraint row A_j, an embedding u_j incorporates:
  - its right-hand side b_j,
  - coefficients $\{A_{j,i}\}$,
  - semantic type (flow, capacity, timing, coupling).
- **Dual violation signals** During inference, RFM computes:

  $$v = \max(0, Ax - b),$$

  which becomes an additional feature injected into attention heads.

Together, these embeddings provide a structured view of the MILP to the downstream components.

5.3 MILP-Transformer: Surrogate Solver Core

The MILP-Transformer is the iterative reasoning engine of RFM. It transforms embeddings using multi-head attention modified to:

- incorporate dual signals $A^\top v$,
- account for constraint–variable interactions,
- refine binary decisions through soft relaxation.

A single block updates x via:

$$x^{(t+1)} = \text{Refine}\Big(\text{MILPAttn}\big(x^{(t)}, A^\top v^{(t)}\big)\Big),$$

where:

- MILPAttn is attention augmented with MILP structure,
- $v^{(t)} = \max(0, Ax^{(t)} - b)$ represents constraint violations,
- Refine performs a latent proximal-like update pushing variables toward feasibility or integrality.

Multiple layers approximate an unrolled primal–dual optimization trajectory. The transformer is not solving the MILP exactly but providing an amortized, differentiable surrogate solver.

5.4 Constraint-Specialized Mixture-of-Experts

Constraints in routing MILPs fall into families:

- flow conservation constraints,
- capacity constraints,
- timing/precedence constraints,
- vehicle or facility assignment constraints,
- logical or binary coupling constraints.

RFM introduces a constraint-specialized MoE module:

$$\Phi = \mathrm{MoE}(v),$$

where each expert specializes in a constraint family. During routing scenarios, different constraints dominate feasibility, enabling the MoE to modulate update directions based on which violations are most relevant.

This enhances robustness and scalability to heterogeneous networks.

5.5 Neural Routing Optimization Model (NROM)

Where the MILP-Transformer performs local refinement, the NROM module provides global reasoning. It integrates:

- graph structure of the routing network,
- constraint embeddings from the MILP-aware encoder,
- variable embeddings from the surrogate solver.

NROM uses graph attention and transformer layers to reason about:

- global connectivity,
- sequencing of routes,
- multi-arc dependencies,
- interactions between different facility types.

The output is a structured routing solution or a high-quality initialization for the MILP-Transformer refinement stage.

5.6 Diffusion Routing Prior

Routing decisions do not arise from a unimodal distribution. Industrial routing systems exhibit multiple feasible modes corresponding to different:

- route structures,
- facility congestion patterns,
- load-balancing strategies,
- fallback or contingency plans.

RFM incorporates a diffusion prior Nichol and Dhariwal [2021], Hoogeboom et al. [2021] over routing variables. In latent space:

$$z_0 \sim p_\theta(z_0) \quad \text{via diffusion sampling,}$$

followed by decoding:

$$x = \text{Decode}(z_0).$$

The diffusion prior serves three purposes:

1. generate diverse feasible initializations,
2. smooth the optimization landscape,
3. regularize the MILP-Transformer refinement.

5.7 Routing World Model

In dynamic routing environments, decisions must anticipate future conditions such as congestion, delays, or demand surges. RFM integrates a world model Ha and Schmidhuber [2018], Hafner et al. [2019] that predicts plausible future states of the logistics network:

$$s_{t+1} = f_\theta(s_t, x_t).$$

The world model enables:

- multi-step planning,
- simulation rollouts,
- cost-to-go prediction,
- sensitivity analysis of routing decisions.

RFM can therefore operate both as a one-shot surrogate solver and as a sequential decision-making system.

5.8 Data Flow and Inference Pipeline

At inference time, RFM performs the following steps:

1. Encode MILP structure (A, b, c) into variable and constraint embeddings.
2. Sample or retrieve an initialization (from NROM or the diffusion prior).
3. Execute MILP-Transformer layers to iteratively refine the solution.
4. Use constraint MoE modules to modulate updates based on violation types.
5. Optionally, perform rollouts using the world model for dynamic optimization.
6. Output a binary routing decision vector $\hat{x}$.

This pipeline mimics classical optimization while remaining fully differentiable and computationally efficient at scale.

5.9 Relation to Classical Solvers and DL4MIP

RFM integrates concepts from:

- primal–dual solvers (dual violation messages $A^\top v$),
- cutting-plane heuristics (constraint experts),
- proximal and projection methods (latent refinement),
- implicit differentiation and differentiable solvers,
- Neural CO and diffusion modeling.

Unlike classical solvers, RFM amortizes computation across instances, making real-time routing feasible.

Unlike pure neural heuristics, RFM respects MILP structure, enabling far stronger generalization and feasibility.

5.10 Summary

The Routing Foundation Model is a fully neural optimization system that:

- embeds MILP structure,
- approximates primal–dual optimization,
- generates diverse routing solutions via diffusion modeling,
- reasons about multi-step dynamics via world models,
- scales to industrial networks while maintaining feasibility.

The next chapter presents the MILP-Transformer module in depth, detailing its architecture, dual-informed attention, proximal refinement, and surrogate solver behavior.

6 Component I: MILP-Transformer

The MILP-Transformer is the core reasoning engine of the Routing Foundation Model (RFM). It performs iterative refinement of routing decisions using transformer layers augmented with MILP structure, dual signals, and proximal-like updates. Whereas the Neural Routing Optimization Model (NROM) provides global structure and initial routing assignments, the MILP-Transformer acts as a learned surrogate solver, approximating primal–dual optimization trajectories through a sequence of differentiable updates.

This chapter introduces the mathematical formulation, architectural design, constraint-aware mechanisms, and pseudocode of the MILP-Transformer.

6.1 Motivation

Classical MILP solvers rely heavily on:

- primal heuristics,
- dual bound propagation,
- constraint violation signals,
- iterative updates that move toward feasibility and improved cost.

Neural CO methods, while flexible, do not incorporate these structural signals directly. The MILP-Transformer bridges this gap by embedding MILP structure into transformer layers, allowing gradient-based models to perform optimization-like updates in the latent space.

6.2 Soft Integer Relaxation

Routing decisions $x \in \{0,1\}^n$ are first relaxed to continuous values:

$$x^{(0)} = \sigma(\ell/\tau),$$

where:

- ℓ are learnable logits,
- σ is a sigmoid activation,
- τ is a temperature controlling softness.

This produces a differentiable surrogate for binary decisions:

$$x \in [0,1]^n.$$

The role of the MILP-Transformer is to iteratively push these continuous values toward feasible, near-integral solutions.

6.3 Constraint Violations and Dual Signals

Given the compact MILP:

$$Ax \leq b, \qquad x \in \{0,1\}^n,$$

we compute the violation vector:

$$v = \max(0, Ax - b),$$

which encodes the magnitude of constraint dissatisfaction.

The dual-inspired message is:

$$h = A^\top v,$$

analogous to a gradient with respect to primal variables in the Lagrangian:

$$\mathcal{L}(x, \lambda) = c^\top x + \lambda^\top (Ax - b).$$

This h signal informs the MILP-Transformer which variables most contribute to constraint violations and should be adjusted.

6.4 MILP-Aware Attention

Standard transformer attention computes:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

For MILP-aware attention, the keys and queries are modified to include dual signals and constraint embeddings:

$$Q = W_Q[x^{(t)}; h], \qquad K = W_K[x^{(t)}; u], \qquad V = W_V x^{(t)},$$

where:

- u is the constraint embedding from the MILP-aware encoder,
- h represents $A^\top v$,
- $[\cdot\,;\,\cdot]$ denotes concatenation.

This modification allows attention to focus on variables most implicated in current constraint violations. Multi-head attention captures interactions between different constraint families (flow, capacity, timing, coupling).

6.5 Constraint Mixture-of-Experts (MoE)

Constraints in routing MILPs vary widely in structure and functional role. To handle this heterogeneity, the MILP-Transformer uses a Mixture-of-Experts gated by violation signals:

$$\Phi = \mathrm{MoE}(v),$$

where experts specialize in:

- flow constraints,
- capacity constraints,
- timing or precedence constraints,
- logical / binary consistency constraints.

The gating mechanism ensures that:

- only relevant experts activate for a given violation pattern,
- updates are constraint-specific,
- training is modular and scalable.

6.6 Latent Gradient / Proximal Refinement

Following MILP-aware attention and MoE updates, the model computes an approximate optimization step:

$$\mathcal{L} = c^\top x + \Phi,$$

$$g = \nabla_x \mathcal{L}.$$

A refinement operator updates the relaxed binary vector:

$$x^{(t+1)} = \mathrm{Refine}\left(x^{(t)}, g\right),$$

where Refine may include:

- projected gradient descent toward $[0, 1]^n$,
- proximal steps encouraging integrality,
- smoothing to avoid oscillations,
- constraint-aware projection based on violation statistics.

This update mimics steps performed by primal heuristics or projected dual methods in classical MILP solvers.

6.7 Overall Forward Pass

A full MILP-Transformer layer performs:

$$x^{(t+1)} = \text{Refine}\Big(\text{MILPAttn}(x^{(t)}, A^\top v^{(t)}), \nabla_x \mathcal{L}(x^{(t)})\Big).$$

Stacking T layers yields a refined solution approaching feasibility and near-integrality.

6.8 Pseudocode

Algorithm 1 MILP-Transformer Surrogate Solver

Input: MILP (A, b, c)
Initialize logits ℓ
$x \leftarrow \sigma(\ell/\tau)$
for $t = 1$ to T **do**
  $v \leftarrow \max(0, Ax - b)$
  $h \leftarrow A^\top v$
  $x \leftarrow \text{MILPAttn}(x, h)$
  $\Phi \leftarrow \text{MoE}(v)$
  $\mathcal{L} \leftarrow c^\top x + \Phi$
  $g \leftarrow \nabla_x \mathcal{L}$
  $x \leftarrow \text{Refine}(x, g)$
end for
Return: x (optionally discretized)

6.9 Interpretation as an Optimization Algorithm

The MILP-Transformer approximates:

- primal descent (via refinement),
- dual ascent (via $A^\top v$),
- constraint filtering (via MoE),
- proximal regularization (via soft relaxation),

- iterative convergence (via stacked layers).

Thus, it behaves like a learned unrolled solver specialized for routing constraints, with dramatically lower inference cost than classical MILP solvers.

6.10 Summary

The MILP-Transformer is a differentiable, optimization-inspired neural surrogate solver capable of:

- encoding constraint violations,
- performing dual-informed attention,
- applying constraint-specific expert reasoning,
- refining routing decisions via proximal updates,
- approaching feasible binary solutions.

The next chapter introduces Component II: the Neural Routing Optimization Model (NROM), which provides global structural reasoning and integrates with the MILP-Transformer to form the complete Routing Foundation Model.

7 Component II: Neural Routing Optimization Model (NROM)

The Neural Routing Optimization Model (NROM) provides the global reasoning component of the Routing Foundation Model (RFM). Whereas the MILP-Transformer focuses on local constraint satisfaction and primal–dual refinement, NROM captures the global structure of routing problems, the geometry of the underlying logistics network, and the high-level interactions between routes, facilities, and constraints.

NROM acts as a structured encoder–decoder system that generates high-quality routing assignments, initializes the MILP-Transformer, and supplies global contextual signals about the network. This chapter formalizes the NROM architecture and its role within RFM.

7.1 Motivation

Classical MILP formulations represent routing decisions in algebraic form, but the underlying logistics network has rich combinatorial structure:

- graph connectivity between facilities,
- vehicle flow patterns,
- precedence and timing dependencies,
- global load-balancing relationships.

The MILP-Transformer alone cannot capture this global geometry because its refinement steps operate primarily in variable space. NROM provides:

1. structural embeddings of the routing network,
2. high-level reasoning over multi-hop dependencies,
3. initialization of the relaxed binary vector x,
4. integration of network topology with MILP structure.

7.2 Formal Definition

Given a routing network represented as a directed graph:

$$G = (V, E),$$

with associated features:

- node types (FC, SC, DS, hubs),
- arc costs, capacities, or distances,
- timing windows or precedence relations,

the NROM computes:

$$z = \text{Encode}(G, A, b, c),$$

and predicts an initial relaxed routing vector:

$$x^{(0)} = \text{Decode}(z).$$

This vector is fed into the MILP-Transformer for constraint-aware refinement.

7.3 Variable and Constraint Embeddings

NROM reuses and extends embeddings from the MILP-aware encoder to produce a unified latent representation.

Variable embeddings. For each MILP variable x_i corresponding to an arc or decision:

$$e_i = f_\theta(c_i, A_{:,i}, \text{node features}, \text{arc features}).$$

These embeddings reflect:

- cost contributions,
- constraint participation,
- network semantics (e.g., FC→SC vs DS→customer arcs),
- geometric attributes (distance, travel time).

Constraint embeddings. For each constraint row:

$$u_j = g_\theta(b_j, A_{j,:}, \text{constraint type}).$$

These embeddings encode:

- flow conservation structure,
- capacity and timing rules,
- MILP families relevant to routing.

7.4 Graph / Sequence Views of Routing Decisions

Routing decisions can be represented in multiple views:

Graph view. RFM treats x as a subset of edges in G defining paths, tours, or flows. NROM uses graph neural networks (GNNs) or Graphormer-style transformers Ying et al. [2021] to propagate structural information across the network.

Sequence view. For problems like TSP, VRP, or courier routes, decisions correspond to ordered sequences. NROM may adopt autoregressive or pointer-network decoders Vinyals et al. [2015], Bello et al. [2017] in addition to graph layers.

Hybrid view. Industrial routing often requires both:

- global graph reasoning (capacities, connectivity),
- local sequencing (within a vehicle route).

NROM integrates both through multi-view encoders.

7.5 Message Passing and Attention Mechanisms

NROM aggregates global information using graph attention:

$$h_v^{(t+1)} = \text{Attn}\Big(h_v^{(t)}, \{h_u^{(t)} : u \in \mathcal{N}(v)\}\Big),$$

where h_v are node embeddings and $\mathcal{N}(v)$ are neighbors.

Similarly, variable embeddings e_i (corresponding to arcs (u, v)) participate in structured attention:

$$e_i^{(t+1)} = \text{GraphAttn}\big(e_i^{(t)}, h_u^{(t)}, h_v^{(t)}\big).$$

This ensures that routing decisions incorporate network geometry, congestion, and multi-hop dependencies.

7.6 Decoder: Generating Initial Routing Assignments

After global reasoning, NROM generates an initial relaxed solution:

$$x^{(0)} = \sigma(We),$$

where e are the refined variable embeddings. The decoder may also produce:

- route orderings (via autoregressive decoding),
- vehicle-load predictions,
- timing feasibility predictions.

This initial solution serves two purposes:

1. it reduces the search space for the MILP-Transformer,
2. it produces high-quality warm starts analogous to MILP heuristics.

7.7 Training Objectives

NROM is trained jointly with the MILP-Transformer using a mixture of objectives:

1. Feasibility loss. Given ground-truth feasible solutions $x^\star$:

$$\mathcal{L}_{\text{feas}} = \|Ax^{(0)} - b\|_+.$$

2. Cost loss.

$$\mathcal{L}_{\text{cost}} = c^\top x^{(0)}.$$

3. Structural reconstruction loss. Predicting arc usage or flow patterns from ground truth.

4. Diffusion-prior consistency. Matching statistics from the diffusion routing prior Nichol and Dhariwal [2021], Hoogeboom et al. [2021].

5. Auxiliary world-model alignment. Aligning initial routing decisions with world-model forecast dynamics Ha and Schmidhuber [2018], Hafner et al. [2019].

These multi-objective losses shape NROM into a globally aware routing initializer.

7.8 Interaction with the MILP-Transformer

1. NROM computes global features and produces $x^{(0)}$.
2. The MILP-Transformer performs local refinement:

$$x^{(T)} = \text{MILPTransformer}(x^{(0)}, A, b, c).$$

3. NROM may provide structural feedback during refinement, such as updated graph embeddings or congestion predictions.

The two components operate analogously to:

- a global heuristic (NROM),

- followed by a local primal–dual solver (MILP-Transformer).

7.9 Summary

NROM provides the global structure and high-level reasoning necessary for large-scale routing optimization. Its integration of graph representations, MILP embeddings, and multi-view attention enables it to:

- encode complex logistics networks,
- generate high-quality initial routing assignments,
- support refinement by the MILP-Transformer,
- generalize across network sizes and topologies.

NROM and the MILP-Transformer together form the core of the Routing Foundation Model.

The next chapter introduces diffusion priors for routing, which allow RFM to model distributional uncertainty, generate diverse feasible solutions, and regularize optimization trajectories.

8 Component III: Diffusion Priors for Routing

Routing problems exhibit inherently multi-modal decision landscapes: multiple distinct route structures, assignment patterns, and load balancing strategies may all produce feasible and near-optimal solutions. Classical MILP solvers approximate this landscape with branch-and-bound search, while neural methods often collapse to a single mode. Diffusion models provide a principled generative framework capable of modeling diverse routing solutions and serving as a learned prior over the decision space.

This chapter introduces diffusion priors for routing, describes their integration with the Routing Foundation Model (RFM), and explains how they improve feasibility, generalization, and robustness.

8.1 Motivation

Unlike smooth continuous spaces, the routing solution space is:

- **highly discrete** (binary MILP variables),
- **combinatorial** (solution space scales exponentially),
- **multi-modal** (many equally good or feasible routes),
- **sensitive to global structure** (graph connectivity, timing),
- **irregular** (constraints vary significantly by facility type).

A generative prior is desirable because it can:

1. provide diverse initializations for the MILP-Transformer,
2. capture global structural invariances (e.g., route symmetries),
3. improve robustness to novel topologies,
4. regularize optimization to avoid poor local minima,
5. align training with distributions of real logistics networks.

Diffusion models Nichol and Dhariwal [2021], Hoogeboom et al. [2021] satisfy these requirements and extend naturally to both continuous and discrete routing settings.

8.2 Latent Representations for Routing

Let $x \in \{0,1\}^n$ denote routing decisions. These variables are embedded into a continuous latent representation z:

$$z = f_\theta(x),$$

where f_θ is an encoder such as a learnable linear map, an MLP, or a graph encoder conditioned on network topology.

The diffusion process acts in latent space:

$$z_0 \in \mathcal{Z}, \qquad x = f_\theta^{-1}(z_0),$$

because:

- continuous diffusion is stable and well-understood,
- latent structure can encode combinatorial geometry,
- decoding enables feasibility-oriented reconstruction.

8.3 Forward Diffusion Process

The forward diffusion process gradually corrupts the latent routing vector z_0:

$$q(z_t \mid z_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\, z_{t-1},\ (1-\alpha_t) I)\,,$$

or in the simplified DDPM formulation:

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

with a variance schedule following Nichol and Dhariwal [2021].

In discrete variants (binary or categorical variables), we may use:

- mask-based corruption,
- bit-flip diffusion,
- discrete autoregressive diffusion Hoogeboom et al. [2021].

8.4 Reverse Diffusion and Routing Priors

The reverse diffusion model learns:

$$p_\theta(z_{t-1} \mid z_t),$$

parameterized by a neural network (typically a transformer or graph network).

At inference, sampling proceeds backward:

$$z_{t-1} = g_\theta(z_t, t),$$

until we produce a clean latent:

$$z_0 = g_\theta(z_1, 1).$$

The routing solution is then decoded:

$$x = \text{Decode}(z_0),$$

and optionally refined using the MILP-Transformer.

8.5 Conditioning Diffusion on MILP Constraints

Unconditional generation may produce infeasible or irrelevant routes. Therefore, the diffusion prior is conditioned on MILP structure:

$$p_\theta(z_{t-1} \mid z_t, A, b, c).$$

Conditioning may include:

- constraint embeddings u_j,
- dual violation signals $A^\top v$,
- cost vector c,
- node and arc features,
- NROM-provided global representations.

This transforms the diffusion process into a structured prior aligned with routing MILPs.

8.6 Feasibility-Aware Denoising

During reverse diffusion, the denoising network may incorporate feasibility projections:

$$z_{t-1} = g_\theta(z_t, t) - \eta_t \nabla_z \|Ax - b\|_+,$$

where $x = \text{Decode}(z_{t-1})$.

This introduces:

- constraint biasing,

- soft feasibility enforcement,
- implicit primal–dual reasoning,
- gradient-like corrections to latent space.

Such feasibility-aware diffusion blends generative modeling with classical constraint satisfaction.

8.7 Integration with NROM and MILP-Transformer

The diffusion prior interacts with the two main RFM components:

(1) With NROM. Diffusion outputs provide:

- diverse global routing hypotheses,
- structural variations for training augmentation,
- noise-injected versions of feasible routes.

NROM may also provide conditioning signals for the diffusion model.

(2) With MILP-Transformer. Diffusion-derived $x^{(0)}$ serves as:

- high-quality initialization,
- exploration mechanism for escaping local minima,
- regularizer preventing premature convergence,
- prior over feasible modes of the routing landscape.

8.8 Training Objectives

The diffusion model is trained with the simplified loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0,t,\epsilon}\left[\|\epsilon - \epsilon_\theta(z_t, t, A, b, c)\|^2\right],$$

optionally augmented by:

- **structure consistency** with graph features,
- **feasibility bias** from $Ax \leq b$,
- **multi-modal diversity** constraints,
- **RL-style value guidance** from world models.

8.9 Summary

Diffusion priors enrich RFM by enabling:

- multi-modal routing solution modeling,

- diverse and robust initializations,
- feasibility-aware generative sampling,
- structured conditioning on MILP constraints,
- smoother optimization landscapes,
- improved generalization across networks.

They form a generative backbone that complements the MILP-Transformer and NROM, driving both exploration and regularization within the RFM pipeline.

The next chapter introduces routing world models, which extend RFM to sequential decision-making and multi-step planning.

9 Component IV: Routing World Models

Modern logistics systems operate in dynamic environments: traffic conditions evolve, demand fluctuates, processing delays accumulate, vehicle availability changes, and local constraints propagate throughout the network over time. Routing decisions therefore cannot be considered in isolation; they must anticipate the downstream effects of actions. World models provide a mechanism for learning these environment dynamics from data and integrating them into the Routing Foundation Model (RFM).

This chapter introduces routing world models, describes their integration with NROM and the MILP-Transformer, and introduces a structured approach using state-space models (SSMs), including Mamba-style selective state space architectures.

9.1 Routing as Sequential Decision-Making

At each decision step t, the routing system observes a state:

$$s_t = (G_t, d_t, c_t, \text{cong}_t, \text{inv}_t, \text{slack}_t),$$

where:

- G_t: network topology and available links,
- d_t: demand or shipment distribution,
- c_t: dynamic edge costs (time, distance, congestion),
- cong_t: congestion indicators at facilities,
- inv_t: inventory / capacity state,
- slack_t: SLA slack or timing remaining.

A routing decision x_t (binary MILP variables or soft relaxations) updates the environment:

$$s_{t+1} = f_\theta(s_t, x_t),$$

where f_θ is learned via a world model.

The world model enables:

- anticipatory decisions,
- multi-step planning and rollouts,
- future constraint prediction,
- routing under uncertainty.

9.2 Latent World Models

Following the latent dynamics frameworks of Ha and Schmidhuber [2018], Hafner et al. [2019], we introduce a latent representation:

$$z_t = \text{Encode}(s_t),$$

and learn a transition model:

$$z_{t+1} = g_\theta(z_t, x_t) + \epsilon_t,$$

where ϵ_t captures stochastic fluctuations.

The decoder reconstructs observable signals:

$$\hat{s}_{t+1} = \text{Decode}(z_{t+1}).$$

For routing, z_t may encode:

- queue lengths,
- travel-time distributions,
- predicted delays,
- congestion spillover patterns,
- facility utilization,
- upcoming SLA violations.

9.3 State Space Models for Logistics Dynamics

World model transitions involve:

- long time horizons (12–48 hours),
- event-driven updates (arrivals, departures),
- continuous temporal dynamics (congestion),
- structured dependencies across facilities.

Transformers struggle with extremely long contexts due to $O(T^2)$ attention cost. State space models (SSMs) instead provide:

$$\dot{h}(t) = Ah(t) + Bu(t), \qquad y(t) = Ch(t),$$

discretized into:

$$h_{t+1} = \bar{A}h_t + \bar{B}u_t, \qquad y_t = Ch_t,$$

where:

- h_t is the latent state,
- u_t contains routing decisions x_t and exogenous inputs.

Such models capture:

- temporal smoothing,
- gradual congestion buildup,
- facility-level inertial effects,
- multi-hop, delayed consequences of routing.

9.4 Mamba-Style Selective State Space Models

Mamba Gu et al. [2024] introduces selective SSMs that modulate updates based on input relevance:

$$h_{t+1} = \bar{A}(x_t)h_t + \bar{B}(x_t)u_t.$$

This "selective" mechanism is ideal for logistics:

- routing decisions affect only local regions of the network,
- not all constraints or facilities should update at each step,
- congestion signals propagate selectively, not globally.

The Mamba-style world model performs:

$$z_{t+1} = g_\theta(z_t, x_t) = \mathrm{SSM}_\theta(z_t, x_t) + \mathrm{Skip}(z_t),$$

with routing-aware gates:

$$g_t = \sigma(W_g[z_t; x_t]).$$

This allows the world model to:

- track long-horizon effects of routing,
- propagate congestion signals efficiently,
- scale to millions of decision steps,
- maintain stability across large temporal windows.

9.5 Predicting Congestion, SLA Violations, and Delays

The world model predicts key routing metrics:

Congestion:

$$\hat{\text{cong}}_{t+1} = C_{\text{cong}} z_{t+1}.$$

Travel times:

$$\hat{\tau}_{uv,t+1} = C_{\tau} z_{t+1}.$$

SLA slack:

$$\hat{\text{slack}}_{t+1} = C_{\text{slack}} z_{t+1}.$$

These predictions allow planners to choose routing assignments that avoid future bottlenecks.

9.6 Planning via Imagination Rollouts

Using the learned transitions:

$$z_{t+1} = g_{\theta}(z_t, x_t),$$

RFM can simulate multiple hypothetical routing strategies:

$$z^{(i)}_{t+k} = g_{\theta}(z^{(i)}_{t+k-1}, x^{(i)}_{t+k-1}), \qquad i = 1, \ldots, M.$$

Rollouts enable:

- evaluating long-term effects of routing,
- selecting actions that minimize expected congestion,
- sensitivity analysis to demand fluctuations,
- scenario planning (e.g., traffic surges).

This "imagination" capability parallels that of Dreamer/Planet Hafner et al. [2019] applied to supply-chain dynamics.

9.7 Interaction with MILP-Transformer and NROM

With the MILP-Transformer:

- world model predicts constraint violations before they occur,
- MILP-Transformer refines solutions given predicted violations,
- dual signals may incorporate future-state penalties.

With NROM:

- NROM incorporates world-model latent states into global embeddings,
- world-model predictions shape the global route structure,
- multi-step forecasts inform initial routing $x^{(0)}$.

This makes RFM a *temporally aware* surrogate solver.

9.8 Training Objectives

World models are trained with a mixture of losses:

Reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \|s_{t+1} - \hat{s}_{t+1}\|.$$

Dynamics prediction loss:

$$\mathcal{L}_{\text{dyn}} = \|z_{t+1} - g_\theta(z_t, x_t)\|.$$

Routing-alignment loss: Encourages consistency with MILP-Transformer and NROM predictions.

SLA / congestion penalty:

$$\mathcal{L}_{\text{SLA}} = \max(0, \hat{\text{slack}} < 0).$$

9.9 Summary

Routing world models enable RFM to reason over time. Using Mamba-style SSMs, they provide:

- long-horizon predictions of congestion and delays,
- anticipatory routing decisions,
- efficient simulation of alternative routing strategies,
- structured selective updates aligned with logistics behavior.

In combination with NROM and the MILP-Transformer, world models turn RFM from a static optimizer into a dynamic, predictive, and adaptive routing foundation model.

10 Training, Evaluation, and Benchmarks

This chapter describes the training methodology for each component of the Routing Foundation Model (RFM), the benchmark datasets used for evaluation, and the metrics employed to assess performance across routing tasks of varying scale and complexity. The goal of this chapter is to establish a reproducible framework for large-scale routing experiments that compare neural surrogate solvers, diffusion priors, world-model architectures, and mixed neural–MILP baselines.

10.1 Benchmark Datasets

We evaluate RFM on two primary classes of routing tasks: synthetic routing benchmarks and industrial-style middle-mile networks.

10.1.1 Synthetic Routing Benchmarks (50–200 Nodes)

These instances are constructed using random geometric graphs with:

- $n \in \{50, 100, 200\}$ nodes,
- Euclidean distances for arc costs,
- heterogeneous vehicle capacities,
- stochastic demand profiles per node,
- optional time windows with soft feasibility penalties.

MILP formulations include:

- flow conservation constraints,
- capacity constraints,
- distance/time budgets,
- subtour-elimination approximations.

10.1.2 Industrial Topologies

The industrial-style benchmarks follow hierarchical network templates inspired by real-world logistics systems:

- facilities of types $\{\text{FC}, \text{SC}, \text{DS}, \text{XD}\}$,
- structured connectivity ($\text{FC} \to \text{SC} \to \text{DS}$),
- realistic inter-facility distances,
- variable congestion patterns and temporal dynamics.

Each instance yields an MILP with hundreds to thousands of binary decision variables.

10.2 Training Procedures

RFM training consists of multiple specialized modules trained either independently or jointly depending on the experiment.

10.2.1 MILP-Transformer Training

The MILP-Transformer is trained to approximate primal–dual optimization via refinement iterations. A typical training iteration consists of:

1. Sampling an MILP instance and constructing variable and constraint embeddings.
2. Running the MILP-Transformer for T refinement steps.
3. Computing losses:

$$\mathcal{L}_{\text{MT}} = c^\top x + \lambda_{\text{feas}} \|Ax - b\|_+ + \lambda_{\text{prox}} \|x - x_{\text{prev}}\|^2.$$

4. Updating parameters with AdamW (lr = 1e$-$4).

The refinement schedule uses a decaying soft-relaxation temperature:

$$\tau_t = \tau_0 \cdot \gamma^t.$$

10.2.2 NROM Training

The Neural Routing Optimization Model (NROM) is trained via multi-objective learning, combining cost, feasibility, and structural regularization:

$$\mathcal{L}_{\text{NROM}} = \mathcal{L}_{\text{cost}} + \lambda_1 \mathcal{L}_{\text{feas}} + \lambda_2 \mathcal{L}_{\text{struct}}.$$

Data augmentations include:

- randomly masking edges,
- perturbing demand and cost vectors,
- reshuffling node roles,
- constraint randomization to improve generalization.

10.2.3 Diffusion Prior Training

The diffusion prior uses the improved DDPM objective from Nichol and Dhariwal [2021]. Training pairs consist of:

- feasible MILP solutions,
- heuristic warm starts,
- diverse near-feasible samples.

The loss is:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0,t,\epsilon}\|\epsilon - \epsilon_\theta(z_t, t, A, b, c)\|^2.$$

Conditioning is provided via:

- constraint embeddings,
- dual violation signals,
- facility-type embeddings.

10.2.4 World Model Training

The world model uses Mamba-based selective SSM layers Gu et al. [2024] to predict future congestion, travel times, inventory levels, and SLA slack under routing decisions.

Training sequences contain:

$$(s_t, x_t, s_{t+1}),$$

where s_t encodes network conditions and x_t represents routing actions.

Loss:

$$\mathcal{L}_{\text{wm}} = \mathcal{L}_{\text{rec}} + \alpha_1\mathcal{L}_{\text{dyn}} + \alpha_2\mathcal{L}_{\text{SLA}}.$$

10.3 Evaluation Metrics

10.3.1 Optimality Gap

For any method producing solution x, the optimality gap is:

$$\text{Gap}(x) = \frac{c^\top x - c^\top x^\star}{c^\top x^\star}.$$

We report:

- mean gap over the test set,
- median gap,
- 95th percentile gap.

10.3.2 Feasibility Violation

Feasibility is measured as:

$$\text{Violation}(x) = \|Ax - b\|_+,$$

with breakdown by:

- flow constraints,
- capacity constraints,
- time constraints,
- integrality violations.

10.3.3 Latency

We measure:

- per-instance inference latency,
- per-layer MILP-Transformer runtime,
- batched throughput,
- CPU vs GPU latency differences.

Latency budgets range from 10ms to 500ms depending on the application.

10.3.4 Generalization

We evaluate generalization across:

- unseen topologies,
- varying node counts,
- perturbed cost structures,
- shifted congestion/time dynamics,
- different MILP formulations (e.g., depot changes).

10.3.5 Ablations

We isolate the contribution of each architectural component by removing:

- dual violation signals,
- constraint MoE,
- diffusion priors,
- NROM warm starts,
- Mamba dynamics modules.

10.4 Summary

This chapter presented the training pipelines, datasets, and evaluation methodology for RFM. The combination of diverse routing benchmarks, multi-component training, and rigorous metrics provides a robust framework for assessing neural routing architectures and guiding future research into large-scale neural optimization.

11 Related Work

The Routing Foundation Model (RFM) builds on several major research directions: neural combinatorial optimization, machine learning for mixed-integer programming, differentiable optimization layers, diffusion models for structured decision-making, and world models for sequential reasoning. This chapter surveys prior work most closely related to the components of RFM.

11.1 Neural Combinatorial Optimization

Neural approaches to routing began with pointer networks Vinyals et al. [2015], which demonstrated that attention-based sequence models could learn heuristics for NP-hard problems like TSP. Subsequent work introduced reinforcement-learning-based decoders for routing Bello et al. [2017], Nazari et al. [2018], and transformer architectures that significantly improved solution quality and generalization Kool et al. [2019].

Despite strong empirical performance, these models typically:

- treat routing as a sequence generation problem,
- lack explicit representation of MILP constraints,
- struggle to generalize beyond training distributions,
- provide no guarantees of feasibility.

Graph-based variants, such as Graphormer-style architectures Ying et al. [2021], introduced structural inductive biases, but they still operate outside the algebraic structure of MILPs. RFM extends this line of work by embedding MILP coefficients, dual signals, and constraint geometry directly into the transformer computations, bridging neural sequence models and classical optimization.

11.2 Machine Learning for MILPs and the DL4MIP Paradigm

The DL4MIP community aims to accelerate or augment classical MILP solvers using learned components. Examples include learning to branch, cut, or warm start using graph neural networks Gasse et al. [2019], Nair et al. [2020]. Surveys such as Bengio et al. [2021] highlight opportunities for hybrid learning–optimization pipelines.

However, these methods:

- depend heavily on existing solver pipelines,
- focus on local components (e.g., branching decisions),
- do not replace the full solver loop,
- offer limited differentiability and amortization.

RFM takes a complementary approach: rather than learning to imitate components of MILP solvers, it learns an end-to-end *surrogate solver* that approximates primal–dual refinement in a fully differentiable, amortized fashion suitable for large-scale routing.

11.3 Differentiable Optimization Layers

Differentiable convex optimization layers such as OptNet Amos and Kolter [2017] and differentiable QP solvers introduced the idea of embedding optimization problems inside deep networks. Later work extended these ideas to structured prediction and bilevel optimization.

While influential, these approaches generally support:

- convex or relaxed problems,
- relatively small-scale instances,
- limited combinatorial structure.

RFM differs in that it addresses large-scale routing MILPs with tens of thousands of binary variables, combining differentiable surrogate solvers with learned constraint-aware refinement and generative priors.

11.4 Diffusion Models for Structured Decision-Making

Diffusion models Nichol and Dhariwal [2021] have transformed generative modeling in continuous domains, and recent variants support discrete structures Hoogeboom et al. [2021]. Their ability to capture multi-modal distributions makes them attractive for routing, where many feasible configurations exist.

Prior work has explored diffusion for graphs, but not in the context of explicit MILP constraints. RFM introduces a *feasibility-conditioned* diffusion prior that interacts with MILP structure, NROM embeddings, and dual signals from the MILP-Transformer. This yields generative initializations aligned with constraint geometry and routing semantics.

11.5 World Models and Sequential Prediction

Latent world models Ha and Schmidhuber [2018], Hafner et al. [2019] have been effective in reinforcement learning domains, enabling agents to perform rollouts in learned latent spaces. These models capture temporal dynamics, uncertainty, and long-horizon dependencies.

Routing environments share these temporal characteristics: congestion, vehicle flow, processing delays, and SLA slack evolve over time. RFM extends world-modeling to logistics by using:

- latent dynamics for supply-chain state transitions,
- prediction of congestion and demand fluctuations,
- multi-step forecasting for routing feasibility,
- integration with MILP-Transformer refinement loops.

11.6 State Space Models and Mamba

State space models (SSMs) provide linear-time sequence processing and long context windows, making them attractive for large temporal forecasting tasks. Mamba Gu et al. [2024] introduced selective SSMs that dynamically modulate updates based on input relevance.

This selective update mechanism is well suited for logistics networks, where only small subsets of nodes or arcs change at each time step. RFM adapts Mamba-style SSMs to form *Routing World Models* capable of:

- predicting facility-level congestion,
- tracking multi-hop flow propagation,
- forecasting SLA violations,
- supporting long-horizon decision planning.

11.7 Summary

RFM sits at the intersection of several important research areas: neural combinatorial optimization, MILP-aware learning, differentiable optimization, diffusion generative models, and world-model-based planning. Its novelty lies in unifying these threads into a single architecture that:

- explicitly embeds MILP structure,
- performs surrogate primal–dual refinement,
- generates diverse routing solutions via diffusion,
- anticipates multi-step dynamics via world models,
- scales to realistic industrial routing networks.

This hybrid view—neural architectures informed by optimization principles—defines the central contribution of RFM.

12 Future Work

The Routing Foundation Model (RFM) establishes a unified framework for neural surrogate optimization, generative routing priors, and world-model–based planning. While the current formulation demonstrates the feasibility of integrating MILP-aware architectures with modern sequence and graph models, several promising research directions remain open. This chapter outlines key extensions that may significantly advance the scalability, robustness, and long-horizon capabilities of neural routing systems.

12.1 World Models for Multi-Step Logistics Planning

Routing decisions propagate through time: congestion accumulates, travel times shift, facilities saturate, and SLA slack tightens. Presenting the problem purely as a static MILP obscures these dynamic interactions. A powerful extension is to develop **learned world models** that predict future logistics states conditioned on candidate routing decisions.

Such models would learn transition dynamics of the form:

$$s_{t+1} = f_\theta(s_t, x_t),$$

where s_t includes congestion maps, facility loads, travel-time distributions, and SLA slack profiles.

Recent selective SSM architectures such as Mamba Gu et al. [2024] are well-suited for this purpose due to their ability to model long sequences with linear-time complexity and selective memory mechanisms. Coupling a world model with a MILP surrogate enables:

- **long-horizon planning** via model-based rollouts,
- **decision refinement** using predicted congestion futures,
- **anticipatory routing** that optimizes for future—not only current—network states,
- **MuZero-style planning** with differentiable surrogate rewards.

An end-to-end RFM planner integrating MILP surrogates and world models could optimize entire daily transportation plans, bridging operations research and model-based reinforcement learning.

12.2 Generative MILP Priors and Large-Scale Pretraining

A central challenge in routing optimization is the combinatorial structure and multi-modality of feasible solutions. Diffusion models already provide a promising foundation for capturing this diversity. A natural next step is to build **large-scale generative priors over MILPs themselves**.

Pretraining over diverse MILPs from:

- vehicle routing,
- matching and assignment,
- scheduling,
- network flow,
- facility-location problems

could yield universal feasibility priors analogous to foundation models in language or vision.

Such pretraining may provide:

- **faster convergence** via improved initialization,
- **zero-shot feasibility** for unseen MILP structures,
- **meta-learning effects** across constraint families,
- **amortized reasoning** for large-scale logistics systems.

Improved diffusion processes Nichol and Dhariwal [2021], Hoogeboom et al. [2021] or flow-based generative models could further accelerate sampling and support conditioning on detailed MILP metadata.

12.3 Hybrid Neural–MILP Pipelines

The MILP-Transformer and NROM provide fast approximate solutions, but classical solvers remain unmatched in final optimality guarantees. A hybrid pipeline leverages the strengths of both worlds:

- neural surrogates generate diverse warm starts,
- feasibility-enforcing refinement reduces constraint violations,
- Gurobi or CPLEX use these warm starts to reduce branch-and-bound search depth,
- latency is dramatically decreased under strict time budgets.

This hybrid approach could achieve:

- near-optimal solutions within milliseconds,
- bounded-optimality regimes for large-scale systems,
- improved reliability across edge cases,
- robustness under structural distribution shifts.

Such pipelines may represent a practical path toward real-world adoption of neural optimization systems in industrial routing environments.

12.4 Graph-Based and SSM-Based Architectures

Routing involves complex spatial and hierarchical relationships that may not be fully captured by generic transformer architectures. Several graph-aware model families offer promising extensions:

Graphormer variants Ying et al. [2021]. These models incorporate centrality encoding, spatial bias, and edge-based attention that naturally encode routing structure.

Selective State Space Models (Mamba). Mamba's capability to model long-range dependencies with linear-time complexity makes it a strong candidate for:

- large networks with long refinement procedures,
- dynamic routing influenced by temporal congestion,
- stability improvements via implicit recurrence.

Hybrid graph + SSM models. Combining graph transformers with SSM recurrence could unify:

- structural reasoning (via attention),
- long-horizon temporal reasoning (via SSMs),
- scalable refinement loops (via selective memory).

Such architectures may yield more stable optimization trajectories, reduced feasibility violations, and improved generalization across network scales.

Summary

The future work directions outlined in this chapter represent broad opportunities for extending the capabilities of routing foundation models. By integrating world-model dynamics, generative MILP priors, hybrid neural–solver pipelines, and advanced architectures, RFM may evolve into a general-purpose optimization framework capable of scaling to the complexity of real-world logistics systems.

13 Discussion and Open Problems

The Routing Foundation Model (RFM) unifies neural surrogate optimization, diffusion generative priors, constraint-aware attention, and world-model reasoning into a single framework for large-scale routing. While the empirical and conceptual foundations are promising, several theoretical, practical, and computational questions remain unanswered. This chapter discusses open challenges, limitations, and potential directions for future research.

13.1 When Can Surrogate Solvers Replace Classical MILPs?

A central question is the extent to which neural surrogate solvers can replace or augment classical MILP solvers. While MILP solvers provide:

- guarantees on optimality gaps,
- deterministic feasibility,
- decades of hand-engineered heuristics,

neural models excel at:

- amortized inference,
- rapid adaptation to new topologies,
- capturing multi-modal solution structures,
- integrating perception, demand prediction, and routing.

Open problems include:

1. Characterizing conditions under which surrogate solvers can reliably produce feasible solutions.
2. Quantifying approximation error and feasibility guarantees.
3. Defining hybrid pipelines that mix neural updates with classical solver callbacks.
4. Developing principled criteria for when to trust neural surrogate outputs in production routing systems.

13.2 Scalability and Robustness

Industrial routing MILPs may contain millions of variables and constraints. Scaling RFM to such sizes requires progress in:

1. Memory efficiency. Transformers and GNNs scale poorly without structural sparsity.

2. Constraint aggregation. Some routing constraints (e.g., flow conservation) appear at massive scale; learning compact representations is nontrivial.

3. Robustness to distribution shift. Routing networks change over time—new facilities open, costs change, traffic patterns shift—which may break the assumptions learned by RFM.

4. Latency guarantees. Real-time logistics requires strict latency budgets (often milliseconds). Neural inference must satisfy worst-case guarantees.

5. Fault tolerance and fallback modes. A production routing system must fail safely when neural predictions are uncertain or infeasible.

13.3 Generalization Across Network Topologies

A significant open problem in Neural CO is generalization. Empirical results suggest:

- transformers generalize poorly to larger graph sizes,
- GNNs struggle with long-range dependencies,
- diffusion models may overfit to specific network structures.

RFM partially addresses this by embedding MILP structure, but deeper questions remain:

1. How to transfer solutions from one geographic region to another?
2. Can we learn invariances over routing structures (e.g., hub-and-spoke, serpentine, mesh networks)?
3. How well does RFM extrapolate to unseen cost regimes or capacity profiles?
4. What architectural or training signals best capture universal routing principles?

13.4 Hybrid Neural–MILP Pipelines

Hybrid pipelines may offer the best of both worlds:

- neural diffusion priors provide initialization,
- NROM offers global structural reasoning,
- MILP-Transformer performs constraint-aware refinement,
- classical solvers finalize feasibility or improve optimality.

Open research questions include:

1. How to integrate learned dual signals into classical solvers?
2. Can neural surrogates serve as warm starts across solver nodes?
3. Can MILP branch-and-bound algorithms incorporate neural priors?

4. What is the right balance between learned components and handcrafted heuristics?

13.5 Learning Theory for Combinatorial Surrogates

A formal theory of surrogate MILP solvers remains elusive:

- What is the sample complexity of learning feasible routing distributions?
- Do diffusion priors converge to MILP-feasible manifolds?
- Under what smoothness assumptions does the MILP-Transformer mimic primal–dual trajectories?
- How do approximation errors propagate across refinement layers?

Establishing theoretical foundations is crucial for reliability in mission-critical logistics environments.

13.6 Diffusion Priors and Multi-Modal Routing

Diffusion models introduce a powerful inductive bias for multi-modal solution spaces, but several open challenges remain:

- Efficiently training diffusion models on high-dimensional routing vectors.
- Conditioning diffusion processes on constraint families without sacrificing diversity.
- Ensuring denoised solutions lie near feasible MILP regions.
- Integrating world-model predictions into diffusion sampling.

Understanding how diffusion priors reshape the routing optimization landscape is an exciting research direction.

13.7 World Models for Logistics and SSM Challenges

Routing world models based on Mamba-style SSMs open new possibilities for anticipatory routing, but also raise new questions:

- How to model multi-timescale effects (minutes vs. hours vs. days)?
- How to represent uncertainty in congestion and demand forecasts?
- How to incorporate exogenous factors (weather, promotions, regional events)?
- Can world-model rollouts be incorporated into MILP refinement as future-value regularizers?
- What is the theoretical stability of SSMs under long-horizon routing updates?

These questions matter for real-world deployment.

13.8 Toward Routing Foundation Models

RFM suggests a broader research agenda:

Universal routing representations. Learned embeddings that transfer across geographic regions, networks, and industrial contexts.

Amortized global optimization. Neural surrogate solvers that handle entire families of related MILPs.

Hierarchical world models. Multi-scale models capturing local, regional, and national routing dynamics.

Unifying routing, forecasting, and planning. Joint models integrating:

- predictive demand modeling,
- congestion forecasting,
- routing optimization,
- risk-aware decision-making.

AGI-oriented perspectives. Routing and logistics represent a rich domain for studying:

- structured reasoning,
- sequential planning,
- constraint satisfaction,
- generalization beyond training data.

Foundation models for optimization may form a key building block for general-purpose reasoning systems.

13.9 Summary

RFM opens a path toward unifying classical optimization with modern deep learning. The opportunities are vast, but so are the challenges: scalability, feasibility guarantees, generalization, theory, and multi-step decision making. Addressing these open problems will be critical for translating foundation-model-based routing into industrial deployments and advancing the frontier of neural optimization research.

14 Conclusion

This monograph introduced the *Routing Foundation Model (RFM)*, a unified neural architecture for large-scale routing and mixed-integer optimization. RFM integrates several complementary components— MILP-aware encoders, the MILP-Transformer surrogate solver, the Neural Routing Optimization Model (NROM), diffusion priors for multi-modal decision landscapes, and Mamba-based world models for sequential logistics dynamics. Together, these modules form a coherent system capable of approximating routing MILPs, anticipating network evolution, and generating diverse, feasible routing strategies with amortized inference cost.

RFM departs from traditional neural combinatorial optimization approaches in several key ways. Rather than treating routing as sequence generation or heuristic imitation, RFM embeds MILP structure directly into its computation: constraint embeddings, dual violation signals, structure-aware attention, and proximal refinement loops allow the model to approximate primal–dual optimization in a differentiable and highly scalable manner. This neural surrogate behavior enables RFM to address industrial routing scenarios where classical solvers face latency constraints or repeated re-optimization demands.

The monograph also highlighted the role of generative modeling in optimization. Diffusion priors, conditioned on MILP structure, allow RFM to capture the inherently multi-modal nature of routing solution spaces and to generate diverse high-quality initializations. Similarly, world models—particularly those based on selective state space architectures— provide a principled mechanism for forecasting congestion, delays, and future constraint conditions, enabling anticipatory and temporally aware routing decisions.

Despite these advances, substantial open challenges remain. Scaling to millions of variables, guaranteeing feasibility, understanding neural approximation of combinatorial structure, and developing theoretical foundations for surrogate MILP solvers represent major research opportunities. The integration of neural and classical optimization continues to be a fertile domain: hybrid pipelines combining diffusion priors, neural surrogate refinement, and classical solver verification may become a practical path toward reliable deployment in supply-chain systems.

More broadly, RFM points toward a future where optimization itself becomes a learned process—amortized, adaptive, and prediction-aware. Routing provides an exceptionally rich testbed for such research, with complex constraints, dynamic environments, and large-scale structure. The techniques explored here may generalize beyond routing to other domains involving planning, resource allocation, control, and structured reasoning.

By unifying optimization principles with neural generative modeling and world-model-based

forecasting, RFM represents one step toward a broader class of *foundation models for decision-making*. The hope is that this framework inspires future work at the intersection of optimization, machine learning, and operations research—advancing both the practical state of industrial routing and the theoretical foundations of neural optimization.

Bibliography

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. *ICML*, 2017.

I. Bello, H. Pham, Q. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv:1611.09940*, 2016.

I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1611.09940.

Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A survey. *European Journal of Operational Research*, 290(2):405–421, 2021.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

M. Blondel, Q. Berthet, M. Cuturi, and S. Vaiter. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2212.07900*, 2022.

X. Chen, Y. Li, L. Song, X. Yao, S. Li, and S. Sun. Learning to solve large-scale routing problems with graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. *NeurIPS*, 2019.

A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2024.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, pages 2555–2565. PMLR, 2019.

E. Hoogeboom, D. Nielsen, J. Peters, R. van den Berg, and M. Welling. Autoregressive diffusion models. In *International Conference on Learning Representations (ICLR)*, 2021.

C. K. Joshi, Q. Cappart, L.-M. Rousseau, P. Veličković, and X. Bresson. Learning tsp requires rethinking generalization. *arXiv preprint arXiv:2006.07054*, 2020.

W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! *ICLR*, 2019.

M. Lodi and G. Zarpellon. Learning to branch in mixed integer programming. *NeurIPS*, 2017.

V. Nair, S. Bartunov, F. Gimeno, et al. Solving mixed integer programs using neural networks. *arXiv:2012.13349*, 2020.

M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. *NeurIPS*, 2018.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR, 2021.

M. Pogančić, M. Paulus, V. Musil, G. Martius, and M. Niepert. Differentiable linear programming layers. *NeurIPS*, 2020.

O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. *NeurIPS*, 2015.

L. Wolsey. *Integer Programming*. Wiley, 1999.

C. Ying, T. Cai, S. Luo, S. Li, G. Li, R. Wang, D. He, and T.-Y. Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

A Additional Derivations

This appendix provides supplementary mathematical derivations that underpin the components of the Routing Foundation Model (RFM). These include primal–dual formulations for routing MILPs, gradient expressions used within the MILP-Transformer, proximal refinement updates, constraint-expert decomposition, diffusion denoising equations, and latent-state evolution for routing world models.

A.1 Primal–Dual Formulation of Routing MILPs

Consider the compact MILP:

$$\min_{x \in \{0,1\}^n} \; c^\top x \quad \text{s.t.} \quad Ax \le b.$$

Relaxing $x \in \{0,1\}^n$ to $x \in [0,1]^n$ yields the LP relaxation:

$$\min_{x \in [0,1]^n} \; c^\top x \quad \text{s.t.} \quad Ax \le b.$$

The Lagrangian is:

$$\mathcal{L}(x, \lambda) = c^\top x + \lambda^\top (Ax - b), \qquad \lambda \ge 0.$$

The dual function is:

$$g(\lambda) = \inf_{x \in [0,1]^n} \mathcal{L}(x, \lambda).$$

Differentiating the constraint term gives the dual update direction:

$$\nabla_\lambda \mathcal{L} = Ax - b,$$

which motivates the violation vector:

$$v = \max(0, Ax - b),$$

and the dual message used in the MILP-Transformer:

$$h = A^\top v.$$

A.2 Derivation of the Soft Integer Relaxation Gradient

Let the relaxed variable be:

$$x_i = \sigma(\ell_i/\tau).$$

Then:

$$\frac{\partial x_i}{\partial \ell_i} = \frac{1}{\tau} x_i(1 - x_i).$$

Thus, gradients from losses propagate to logits as:

$$\frac{\partial \mathcal{L}}{\partial \ell_i} = \frac{1}{\tau} x_i(1 - x_i) \cdot \frac{\partial \mathcal{L}}{\partial x_i}.$$

This induces integrality pressure near 0 or 1 as $x_i(1 - x_i)$ becomes small.

A.3 Proximal Refinement Update

Let the refinement step minimize:

$$\mathcal{R}(x) = \mathcal{L}(x) + \frac{1}{2\eta}\|x - x^{(t)}\|^2,$$

where η is a proximal coefficient.

Setting the derivative to zero:

$$\nabla_x \mathcal{L}(x^{(t)}) + \frac{1}{\eta}(x^{(t+1)} - x^{(t)}) = 0,$$

yields the update:

$$x^{(t+1)} = x^{(t)} - \eta\, \nabla_x \mathcal{L}(x^{(t)}).$$

The MILP-Transformer replaces this with a learned refinement operator:

$$x^{(t+1)} = \text{Refine}(x^{(t)}, g),$$

where $g = \nabla_x \mathcal{L}$.

A.4 MoE Constraint Decomposition

Suppose constraints are partitioned into K families:

$$\{A^{(1)}, A^{(2)}, \ldots, A^{(K)}\}.$$

Let violation components be:

$$v^{(k)} = \max(0, A^{(k)}x - b^{(k)}).$$

The Mixture-of-Experts computes:

$$\Phi = \sum_{k=1}^{K} \alpha_k \, \phi_k(v^{(k)}),$$

where gating weights are:

$$\alpha_k = \text{softmax}(Wv).$$

Gradients through MoE compute:

$$\frac{\partial \Phi}{\partial x} = \sum_{k=1}^{K} \alpha_k \, A^{(k)\top} \nabla \phi_k(v^{(k)}).$$

This corresponds to constraint-family-specific dual updates.

A.5 Diffusion Denoising Derivations

Using the DDPM forward process:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon,$$

the ideal denoiser predicts ϵ:

$$\epsilon_\theta(z_t, t) \approx \epsilon.$$

Solving for z_0 yields:

$$\hat{z}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(z_t, t) \right).$$

The reverse sampling step is:

$$z_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \hat{z}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\, \epsilon'.$$

For discrete routing variables, the diffusion process uses a masked categorical corruption model, whose reverse distribution is:

$$p_\theta(x_{t-1} \mid x_t) = \text{softmax}(W_t h_\theta(x_t, t)).$$

A.6 World Model Latent Dynamics

RFM's world model follows Mamba-style selective state space updates:

Continuous-time SSM:

$$\dot{h}(t) = Ah(t) + Bu(t), \qquad y(t) = Ch(t).$$

Discretization with step Δ:

$$h_{t+1} = \bar{A} h_t + \bar{B} u_t,$$

where:

$$\bar{A} = e^{A\Delta}, \qquad \bar{B} = \int_0^\Delta e^{A(\Delta - \tau)} B \, d\tau.$$

Selective gating adds:

$$h_{t+1} = g_t \odot (\bar{A} h_t + \bar{B} u_t) + (1 - g_t) \odot h_t,$$

where:

$$g_t = \sigma(W_g [h_t; u_t]).$$

This yields long-range propagation of congestion and delay signals.

A.7 MILP Graph Bipartite Embedding

RFM embeds MILP structure using the bipartite graph:

$$\mathcal{G} = (X, C, E),$$

where:

- X = variables,
- C = constraints,
- $E = \{(i, j) : A_{j,i} \neq 0\}$.

A GNN update step is:

$$x_i^{(t+1)} = f_x \left(x_i^{(t)}, \sum_{j:(i,j)\in E} A_{j,i}\, u_j^{(t)} \right),$$

$$u_j^{(t+1)} = f_c \left(u_j^{(t)}, \sum_{i:(i,j)\in E} A_{j,i}\, x_i^{(t)} \right).$$

This provides structural context to MILP-Transformer layers.

A.8 Summary

This appendix presented derivations that support RFM's key components: primal–dual updates, soft relaxation, proximal refinement, MoE constraint decomposition, diffusion denoising, state-space world model dynamics, and MILP bipartite graph embeddings. These foundations clarify how neural reasoning, MILP structure, and generative modeling interact within the RFM architecture.

B Implementation Details

This appendix describes implementation details for the components of the Routing Foundation Model (RFM), including data generation, MILP construction, model configurations, training procedures, optimization hyperparameters, and evaluation protocols. These details support reproducibility and clarify practical considerations in large-scale routing experiments.

B.1 Routing MILP Construction

All routing instances are represented in the compact form:

$$\min_x c^\top x \quad \text{s.t. } Ax \le b,\ x \in \{0,1\}^n.$$

MILPs were generated using:

- realistic hub–spoke and multi-echelon network templates,
- random geometric graphs for synthetic topologies,
- facility types drawn from {FC, SC, DS, cross-dock},
- cost matrices incorporating distance, congestion, and traffic noise.

Flow constraints. For each facility u, flow conservation is encoded as:

$$\sum_v x_{uv} = \sum_v x_{vu}.$$

Capacity constraints. Each vehicle or facility capacity constraint is:

$$\sum_i a_i x_i \le b.$$

Timing / SLA constraints. Travel-time and processing windows are linearized when possible:

$$\sum_{(u,v)} \tau_{uv} x_{uv} \le \text{SLA}.$$

B.2 Data Generation and Preprocessing

Routing datasets include:

- synthetic random geometric networks with 50–200 nodes,
- industrial-style networks with hierarchical facility layouts,
- time-varying congestion profiles sampled from stochastic models,
- demand snapshots sampled from real-like distributions.

Each instance is preprocessed into:

- node features (type, capacity, demand),
- arc features (distance, cost, time),
- constraint embeddings from MILP matrices,
- graph adjacency structures for NROM and world models.

B.3 Model Architectures

B.3.1 MILP-Aware Encoder

Variable embeddings use:

- cost coefficient c_i,
- constraint participation vector $A_{:,i}$,
- node and arc metadata,
- positional or geometric encodings for graph structure.

Constraint embeddings use:

- right-hand side b_j,
- row coefficients $A_{j,:}$,
- constraint-family type,
- dual violation statistics.

Dimensionality:

$$d_{\text{var}} = d_{\text{const}} = 128.$$

B.3.2 MILP-Transformer

Each MILP-Transformer layer contains:

- multi-head MILP-aware attention (8 heads),
- constraint-specific MoE with 4–8 experts,
- latent refinement MLP with GELU activation,
- residual + layer normalization.

Soft-relaxation temperature schedule:

$$\tau_t = 0.5 \cdot 0.95^t.$$

Number of layers:

$$T \in \{6, 8, 10\}.$$

B.3.3 NROM (Neural Routing Optimization Model)

NROM uses a hybrid encoder:

- Graphormer / GAT-style layers for network topology,
- transformer blocks for sequence reasoning,
- cross-attention between variable and node embeddings.

Decoder generates initial relaxed routing decisions:

$$x^{(0)} = \sigma(We),$$

where e are processed variable embeddings.

B.3.4 Diffusion Prior

Latent dimension: 64 or 128.

Forward noise schedule follows Nichol and Dhariwal [2021]. Reverse denoiser uses:

- 8–12 transformer layers,
- conditioning via constraint embeddings,
- optional feasibility projection.

B.3.5 World Model (SSM/Mamba)

The routing world model uses:

- selective SSM layers as in Gu et al. [2024],
- latent dimensionality 64–128,
- GRU-style skip connections for stability,
- cross-attention to integrate routing decisions.

Prediction heads reconstruct:

- congestion cong_{t+1},
- travel times τ_{t+1},
- inventory levels,
- SLA slack.

B.4 Training Procedures

B.4.1 MILP-Transformer Training

Objective:

$$\mathcal{L}_{\text{MT}} = c^\top x + \|Ax - b\|_+ + \beta \cdot \text{Reg}(x).$$

Optimizer:

- AdamW, learning rate 1e$-$4,
- cosine decay schedule,
- gradient clipping at 1.0.

B.4.2 NROM Training

Joint losses:

$$\mathcal{L}_{\text{NROM}} = \mathcal{L}_{\text{feas}} + \lambda_1 \mathcal{L}_{\text{cost}} + \lambda_2 \mathcal{L}_{\text{struct}}.$$

Warm-starting significantly improves convergence.

B.4.3 Diffusion Model Training

Loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0,t,\epsilon} \|\epsilon - \epsilon_\theta(z_t, t, A, b, c)\|^2.$$

Noise levels: 1000 steps.

Augmentations:

- constraint masking,
- edge-drop perturbations,
- topology reshuffling.

B.4.4 World Model Training

Loss:

$$\mathcal{L}_{\text{wm}} = \mathcal{L}_{\text{rec}} + \alpha_1 \mathcal{L}_{\text{dyn}} + \alpha_2 \mathcal{L}_{\text{SLA}}.$$

Training uses:

- AdamW,
- batch size 32–64 trajectories,
- rollout horizon 24–64 steps.

B.5 Evaluation Metrics

RFM is evaluated on:

- **Optimality gap:**
$$\frac{c^\top x - c^\top x^\star}{c^\top x^\star}.$$
- **Feasibility violation:**
$$\|Ax - b\|_+.$$
- **Latency:** end-to-end inference time.
- **Generalization:** performance on unseen network topologies.
- **Diffusion diversity:** entropy of generated routing samples.
- **World-model accuracy:** prediction error of congestion, slack, and dynamics.

B.6 Practical Considerations

Numerical stability. Dual violation vectors may exhibit large magnitudes; normalization is crucial.

Constraint sparsity. A is sparse; efficient sparse attention kernels improve scalability.

Batching. MILP matrices vary in shape; padding and block-diagonal batching were used.

Mixed-precision training. AMP improves training speed without degrading feasibility.

B.7 Summary

This appendix outlined the practical considerations needed to implement RFM at scale. These details—MILP construction, model architectures, training schedules, optimization strategies, and evaluation metrics—form the basis for reproducibility and industrial deployment.

C Extended Experimental Protocols

This appendix provides detailed protocols for generating datasets, training RFM components, running controlled experiments, and evaluating performance across a wide range of routing and MILP settings. These procedures ensure reproducibility and allow systematic comparison of neural surrogate solvers, diffusion priors, and world-model-based planning architectures.

C.1 Routing Instance Generation

We evaluate RFM on two families of routing datasets: synthetic geometric networks and industrial-style multi-echelon logistics networks.

C.1.1 Synthetic 50–200 Node Instances

Nodes are sampled from a 2D Euclidean plane with uniform density. For each instance:

- $n \in \{50, 100, 200\}$ nodes,
- edge distances computed via Euclidean metric,
- demand levels sampled from a truncated normal distribution,
- time windows generated with slack proportional to distance,
- vehicle capacities drawn from a discrete set.

MILP constraints include:

- flow conservation,
- vehicle capacity,
- time-window feasibility,
- subtour elimination approximations.

C.1.2 Industrial Middle-Mile Topologies

We construct networks mimicking Amazon-style routing graphs with:

- facility types: FC, SC, DS, cross-docks,
- hierarchical connectivity: FC$\rightarrow$SC$\rightarrow$DS,
- realistic inter-facility distances,

- stochastic congestion and seasonal demand.

MILPs include thousands of binary variables per instance.

C.2 Baselines

We compare RFM against:

- **Gurobi** (exact solver with tuned heuristics),
- **CPLEX** (canonical MILP solver),
- **Neural CO** pointer-network baselines Bello et al. [2017],
- **Attention-based routing** Vinyals et al. [2015],
- **Graphormer routing models** Ying et al. [2021],
- **Learned branching / cutting** Nair et al. [2020], Gasse et al. [2019],
- **Diffusion-based CO models** Hoogeboom et al. [2021].

All neural baselines use similar parameter counts for fairness.

C.3 Training Pipelines

C.3.1 MILP-Transformer Training

Each training epoch consists of:

1. Sample a routing MILP instance.
2. Construct variable + constraint embeddings.
3. Perform T refinement steps of the MILP-Transformer.
4. Compute losses:
$$\mathcal{L} = c^\top x + \lambda_{\text{feas}} \|Ax - b\|_+ + \lambda_{\text{prox}} \|x - x_{\text{prev}}\|^2.$$
5. Update network parameters via AdamW.

We evaluate convergence in:

- optimality gap,
- feasibility violation,
- stability of refinement steps.

C.3.2 Diffusion Prior Training

We follow the improved DDPM training method of Nichol and Dhariwal Nichol and Dhariwal [2021]. The forward process uses β_t increasing linearly.

Training batches include:

- feasible MILP solutions,
- near-feasible solver warm starts,

- diverse routing samples from heuristics.

Conditioning signal includes:

- constraint embeddings,
- dual violation statistics,
- network topology embeddings.

Metrics:

- negative log-likelihood (optional),
- sample feasibility rate,
- sample diversity score,
- reverse process stability.

C.3.3 NROM Training Protocol

Training uses multi-task loss:

$$\mathcal{L}_{\text{NROM}} = \mathcal{L}_{\text{cost}} + \lambda_1 \mathcal{L}_{\text{feas}} + \lambda_2 \mathcal{L}_{\text{struct}}.$$

Data augmentations:

- topology perturbations,
- synthetic demand shifts,
- constraint randomization,
- cost perturbations.

C.3.4 World Model Training (Mamba-Based)

Datasets consist of network state trajectories:

$$(s_t, x_t, s_{t+1}).$$

Each s_t includes:

- congestion heatmaps,
- predicted arrival times,
- facility utilization,
- SLA slack estimations.

Architecture uses Mamba SSM layers Gu et al. [2024] with:

- kernel length 16–64,
- latent dimension 128,
- selective update gates,
- cross-attention to routing embeddings.

Loss terms:

$$\mathcal{L}_{\text{wm}} = \mathcal{L}_{\text{dyn}} + \alpha_1 \mathcal{L}_{\text{cong}} + \alpha_2 \mathcal{L}_{\text{SLA}}.$$

C.4 Evaluation Protocols

C.4.1 Optimality Gap Evaluation

For each method, we compute:

$$\mathrm{Gap}(x) = \frac{c^\top x - c^\top x^\star}{c^\top x^\star}.$$

We evaluate both:

- mean gap,
- 95th percentile (worst-case) gap.

C.4.2 Feasibility Evaluation

We measure raw violation:

$$\|Ax - b\|_+.$$

And constraint-family breakdown:

- flow conservation,
- capacity,
- timing,
- integrality rounding.

C.4.3 Latency Evaluation

We measure:

- per-instance inference time,
- per-layer MILP-Transformer runtime,
- batching throughput,
- GPU vs CPU speed differences.

Latency budgets target:

$$10\ \text{ms} - 500\ \text{ms}.$$

C.4.4 Generalization Studies

We evaluate:

- unseen network topologies,
- new facility-type mixtures,
- larger-than-training instances,
- temporal shift in congestion models.

C.4.5 Ablation Studies

We ablate:

- removing dual signals,
- removing constraint MoE,
- removing diffusion priors,
- replacing Mamba with GRU/LSTM,
- reduced refinement steps,
- no topology embeddings.

C.5 Summary

This appendix presented comprehensive experimental protocols for evaluating RFM across synthetic and industrial routing tasks. These protocols ensure rigorous comparison of surrogate solvers, generative models, and world-model-based planning modules, enabling reproducible research and large-scale benchmarking of neural optimization systems.